

Contents

1 Routine/Function Prologues	2
1.1 Fortran: Module Interface time_manager.F90 (Source File: time_manager.F90)	2
1.1.1 timemgr_init (Source File: time_manager.F90)	3
1.1.2 timemgr_print (Source File: time_manager.F90)	5
1.1.3 timemgr_print (Source File: time_manager.F90)	6
1.1.4 advance_timestep (Source File: time_manager.F90)	6
1.1.5 get_step_size (Source File: time_manager.F90)	7
1.1.6 get_nstep (Source File: time_manager.F90)	7
1.1.7 get_curr_day (Source File: time_manager.F90)	7
1.1.8 get_prev_date (Source File: time_manager.F90)	8
1.1.9 get_start_date (Source File: time_manager.F90)	8
1.1.10 get_ref_date (Source File: time_manager.F90)	9
1.1.11 get_curr_time (Source File: time_manager.F90)	9
1.1.12 get_curr_calday (Source File: time_manager.F90)	10
1.1.13 is_end_curr_day (Source File: time_manager.F90)	10
1.1.14 is_end_curr_month (Source File: time_manager.F90)	11
1.1.15 is_first_step (Source File: time_manager.F90)	11
1.1.16 is_first_restart_step (Source File: time_manager.F90)	11
1.1.17 is_last_step (Source File: time_manager.F90)	12
1.1.18 timemgr_write_restart (Source File: time_manager.F90)	12
1.1.19 timemgr_read_restart (Source File: time_manager.F90)	12
1.1.20 chkrc (Source File: time_manager.F90)	13

1 Routine/Function Prologues

1.1 Fortran: Module Interface time_manager.F90 (Source File: time_manager.F90)

This module contains wrapper functions that uses the ESMF time manager for time management

INTERFACE:

```
module time_manager
```

```
#if (defined OFFLINE)
```

USES:

```
use precision, only: r8
use ESMF_TimeMgmtMod, only: &
    esmf_errhandlersettype, esmf_err_return, esmf_errprint, esmf_success, &
    esmf_time, esmf_timeinit, esmf_timeget, esmf_timegetdays, &
    esmf_timeincrement, esmf_timedecrement, &
    esmf_date, esmf_dateinit, esmf_gregorian, esmf_no_leap, esmf_dateget, &
    esmf_dateincrementsec, esmf_dateincrementday, esmf_datedecrement, &
    esmf_datediff, esmf_dategetfltdayofyear, &
    esmf_tiememgr, esmf_tiememgrinit, esmf_tiememgradvance, esmf_tiememgrgetnstep, &
    esmf_tiememgrgetstepsize, esmf_tiememgrgetstartdate, esmf_tiememgrgetbasedate, &
    esmf_tiememgrlaststep, esmf_tiememgrgetcurrdate, esmf_tiememgrgetprevdate, esmf_dateisla
    esmf_tiememgrrestartwrite, esmf_tiememgrrestartread
use string_utils, only: to_upper
#if (defined SPMD)
    use spmdMod, only: mpicom, mpiint, masterproc
#else
    use spmdMod, only: masterproc
#endif
```

ARGUMENTS:

```
public :: &
    timemgr_init,                      ! time manager initialization
    advance_timestep,                  ! increment timestep number
    get_step_size,                     ! return step size in seconds
    get_nstep,                         ! return timestep number
    get_curr_date,                     ! return date components at end of current timestep
    get_prev_date,                     ! return date components at beginning of current timestep
    get_start_date,                   ! return date components of the start date
    get_ref_date,                      ! return date components of the reference date
    get_curr_time,                     ! return components of elapsed time since reference date
    get_curr_calday,                  ! return calendar day at end of current timestep
    is_first_step,                     ! return true on first step of initial run
    is_first_restart_step,             ! return true on first step of restart or branch run
```

```

is_end_curr_day,           &! return true on last timestep in current day
is_end_curr_month,         &! return true on last timestep in current month
is_last_step,              &! return true on last timestep
timemgr_write_restart,    &! write info to file needed to restart the time manager
timemgr_read_restart,     &! read info from file needed to restart the time manager
timemgr_restart            ! restart the time manager

Public data for namelist input

character(len=32), public :: &
    calendar = 'GREGORIAN'      ! Calendar in date calculations ('NO_LEAP' or 'GREGORIAN')

integer, parameter :: uninit_int = -999999999 !This is private to this module

integer, public :: &
    dtime      = uninit_int,   &! timestep in seconds
    nestep     = uninit_int,   &! final timestep (or day if negative) number
    nelapse    = uninit_int,   &! number of timesteps (or days if negative) to extend a run
    start_ymd  = uninit_int,   &! starting date for run in yearmmdd format
    start_tod   = 0,          &! starting time of day for run in seconds
    stop_ymd   = uninit_int,   &! stopping date for run in yearmmdd format
    stop_tod   = 0,          &! stopping time of day for run in seconds
    ref_ymd    = uninit_int,   &! reference date for time coordinate in yearmmdd format
    ref_tod    = 0            ! reference time of day for time coordinate in seconds

```

1.1.1 timemgr_init (Source File: time_manager.F90)

Initialize the ESMF time manager.

NOTE - This assumes that the namelist variables have been set before this routine is called.

INTERFACE:

```
subroutine timemgr_init()
```

CONTENTS:

```
!-----
! Initialize error handling.
!-----
call esmf_errhandlersettype(esmf_err_return)
!-----
! Initialize calendar type.
!-----
cal = to_upper(calendar)
if ( trim(cal) == 'NO_LEAP' ) then
    cal_type = esmf_no_leap
else if ( trim(cal) == 'GREGORIAN' ) then
    cal_type = esmf_gregorian
```

```
else
    write(6,*)
    sub,: unrecognized calendar specified: ',calendar
    call endrun
end if
!-----
! Initialize timestep size.
!-----
if ( dtime == uninit_int ) then
    write(6,*)
    sub,: dtime must be specified in namelist'
    call endrun
end if
if ( mod(86400,dtime) /=0 ) then
    write(6,*)
    sub,: timestep must divide evenly into 1 day'
    call endrun
end if

step_size = esmf_timeinit(0, dtime, rc)
call chkrc(rc, sub//': error return from esmf_timeinit: setting step_size')
!-----
! Initialize start date.
!-----
if ( start_ymd == uninit_int ) then
    write(6,*)
    sub,: start_ymd must be specified in namelist'
    call endrun
end if
if ( start_tod == uninit_int ) then
    write(6,*)
    sub,: start_tod must be specified in namelist'
    call endrun
end if
start_date = esmf_dateinit(cal_type, start_ymd, start_tod, rc)
call chkrc(rc, sub//': error return from esmf_dateinit: setting start_date')
!-----
! Initialize reference date for time coordinate.
!-----
if ( ref_ymd /= uninit_int ) then
    ref_date = esmf_dateinit(cal_type, ref_ymd, ref_tod, rc)
else
    ref_date = esmf_dateinit(start_date, rc)
end if
call chkrc(rc, sub//': error return from esmf_dateinit: setting ref_date')
!-----
! Initialize stop date.
!-----
if ( stop_ymd /= uninit_int ) then
    stop_date = esmf_dateinit(cal_type, stop_ymd, stop_tod, rc)
else if ( nestep /= uninit_int ) then
    if ( nestep >= 0 ) then
        stop_date = esmf_dateincrementsec(start_date, dtime*nestep, rc)
```

```

    else
        stop_date = esmf_dateincrementday(start_date, -nestep, rc)
    end if
else if ( nelapse /= uninit_int ) then
    if ( nelapse >= 0 ) then
        stop_date = esmf_dateincrementsec(start_date, dtime*nelapse, rc)
    else
        stop_date = esmf_dateincrementday(start_date, -nelapse, rc)
    end if
else
    write(6,*), 'sub,: Must specify one of stop_ymd, nestep, or nelapse'
    call endrun
end if
call chkrc(rc, sub//': error return setting stop_date')
!-----
! Initialize a time manager.
!-----
tm_id = esmf_tiemngrinit(step_size, start_date, stop_date, ref_date, rc)
call chkrc(rc, sub//': error return from esmf_tiemngrinit')
!-----
! Calculation of ending timestep number (nestep) assumes a constant stepsize.
!-----
ntspday = 86400/dtime
diff = esmf_timeinit()
call esmf_datediff(start_date, stop_date, diff, islater, rc)
call chkrc(rc, sub//': error return from esmf_datediff calculating nestep')
call esmf_timeget(diff, ndays, nsecs, rc)
call chkrc(rc, sub//': error return from esmf_timeget calculating nestep')
nestep = ntspday*ndays + nsecs/dtime
if ( mod(nsecs,dtime) /= 0 ) nestep = nestep + 1
!-----
! Print configuration summary to log file (stdout).
!-----
if (masterproc) then
    call timemgr_print()
end if

```

1.1.2 timemgr_print (Source File: time_manager.F90)

Restart the ESMF time manager.

NOTE - Assumptions: 1) The namelist variables have been set before this routine is called.
The stop date is the only thing that can be changed by the user on a restart.
2) Restart data have been read on the master process before this routine is called.
(timemgr_read_restart called from control/restart.F90::read_restart)

INTERFACE:

```
subroutine timemgr_restart()
```

1.1.3 timemgr_print (Source File: *time_manager.F90*)

Prints the time manager information

INTERFACE:

```
subroutine timemgr_print()
```

CONTENTS:

```
call esmf_tiemngrrestartwrite(tm_id, type, nstep, step_days, step_sec, &
                           start_ymd, start_tod, stop_ymd, stop_tod, ref_ymd, &
                           ref_tod, curr_ymd, curr_tod, rc)
call chkrc(rc, sub//': error return from esmf_tiemngrrestartwrite')

write(6,*), '***** Time Manager Configuration *****'

if ( type == esmf_no_leap ) then
  cal = 'NO_LEAP'
else if ( type == esmf_gregorian ) then
  cal = 'GREGORIAN'
end if

write(6,*), 'Calendar type: ', trim(cal)
write(6,*), 'Timestep size (seconds): ', (step_days*86400 + step_sec)
write(6,*), 'Start date (ymd tod): ', start_ymd, start_tod
write(6,*), 'Stop date (ymd tod): ', stop_ymd, stop_tod
write(6,*), 'Reference date (ymd tod): ', ref_ymd, ref_tod
write(6,*), 'Current step number: ', nstep
write(6,*), 'Ending step number: ', nestep
write(6,*), 'Current date (ymd tod): ', curr_ymd, curr_tod

write(6,*), '*****'
```

1.1.4 advance_timestep (Source File: *time_manager.F90*)

Increment the timestep number.

INTERFACE:

```
subroutine advance_timestep()
```

CONTENTS:

```

call esmf_tiemgradvance(tm_id, rc)
call chkrc(rc, sub//': error return from esmf_tiemgradvance')
!-----
! Set first step flag off.
!-----
first_restart_step = .false.

```

1.1.5 get_step_size (Source File: *time_manager.F90*)

Return the step size in seconds.

INTERFACE:

```
function get_step_size()
```

CONTENTS:

```

call esmf_tiemgrgetstepsize(tm_id, days, seconds, rc)
call chkrc(rc, sub//': error return from esmf_tiemgrgetstepsize')
get_step_size = 86400*days + seconds

```

1.1.6 get_nstep (Source File: *time_manager.F90*)

Return the timestep number.

INTERFACE:

```
function get_nstep()
```

CONTENTS:

```

get_nstep = esmf_tiemgrgetnstep(tm_id, rc)
call chkrc(rc, sub//': error return from esmf_tiemgrgetnstep')

```

1.1.7 get_curr_day (Source File: *time_manager.F90*)

Return date components valid at end of current timestep with an optional offset (positive or negative) in seconds.

INTERFACE:

```
subroutine get_curr_date(yr, mon, day, tod, offset)
```

CONTENTS:

```

date = esmf_tiememgrgetcurrdate(tm_id, rc)
call chkrc(rc, sub//': error return from esmf_tiememgrgetcurrdate')

if (present(offset)) then
  if (offset > 0) then
    date = esmf_dateincrementsec(date, offset, rc)
    call chkrc(rc, sub//': error incrementing current date')
  else if (offset < 0) then
    off = esmf_timeinit(0, -offset, rc)
    call chkrc(rc, sub//': error setting offset time type')
    date = esmf_datedecrement(date, off, rc)
    call chkrc(rc, sub//': error decrementing current date')
  end if
end if

call esmf_dateget(date, ymd, tod, rc)
call chkrc(rc, sub//': error return from esmf_dateget')
yr = ymd/10000
mon = mod(ymd, 10000) / 100
day = mod(ymd, 100)

```

1.1.8 get_prev_date (Source File: *time_manager.F90*)

Return date components valid at beginning of current timestep.

INTERFACE:

```
subroutine get_prev_date(yr, mon, day, tod)
```

CONTENTS:

```

date = esmf_tiememgrgetprevdate(tm_id, rc)
call chkrc(rc, sub//': error return from esmf_tiememgrgetprevdate')

call esmf_dateget(date, ymd, tod, rc)
call chkrc(rc, sub//': error return from esmf_dateget')
yr = ymd/10000
mon = mod(ymd, 10000) / 100
day = mod(ymd, 100)

```

1.1.9 get_start_date (Source File: *time_manager.F90*)

Return date components valid at beginning of initial run.

INTERFACE:

```
subroutine get_start_date(yr, mon, day, tod)
```

CONTENTS:

```
date = esmf_timemgrgetstartdate(tm_id, rc)
call chkrc(rc, sub//': error return from esmf_timemgrgetstartdate')

call esmf_dateget(date, ymd, tod, rc)
call chkrc(rc, sub//': error return from esmf_dateget')
yr = ymd/10000
mon = mod(ymd, 10000) / 100
day = mod(ymd, 100)
```

1.1.10 get_ref_date (Source File: *time_manager.F90*)

Return date components of the reference date.

INTERFACE:

```
subroutine get_ref_date(yr, mon, day, tod)
```

CONTENTS:

```
date = esmf_timemgrgetbasedate(tm_id, rc)
call chkrc(rc, sub//': error return from esmf_timemgrgetbasedate')

call esmf_dateget(date, ymd, tod, rc)
call chkrc(rc, sub//': error return from esmf_dateget')
yr = ymd/10000
mon = mod(ymd, 10000) / 100
day = mod(ymd, 100)
```

1.1.11 get_curr_time (Source File: *time_manager.F90*)

Return time components valid at end of current timestep. Current time is the time interval between the current date and the reference date.

INTERFACE:

```
subroutine get_curr_time(days, seconds)
```

CONTENTS:

```
cdate = esmf_timemgrgetcurrdate(tm_id, rc)
call chkrc(rc, sub//': error return from esmf_timemgrgetcurrdate')

rdate = esmf_timemgrgetbasedate(tm_id, rc)
```

```

call chkrc(rc, sub//': error return from esmf_tiememgrgetbasedate')

call esmf_datediff(rdate, cdate, diff, islater, rc)
call chkrc(rc, sub//': error return from esmf_datediff')

call esmf_timeget(diff, days, seconds, rc)
call chkrc(rc, sub//': error return from esmf_timeget')

```

1.1.12 get_curr_calday (Source File: *time_manager.F90*)

Return calendar day at end of current timestep with optional offset. Calendar day 1.0 = 0Z on Jan 1.

INTERFACE:

```
function get_curr_calday(offset)
```

CONTENTS:

```

date = esmf_tiememgrgetcurrdate(tm_id, rc)
call chkrc(rc, sub//': error return from esmf_tiememgrgetcurrdate')

if (present(offset)) then
  if (offset > 0) then
    date = esmf_dateincrementsec(date, offset, rc)
    call chkrc(rc, sub//': error incrementing current date')
  else if (offset < 0) then
    off = esmf_timeinit(0, -offset, rc)
    call chkrc(rc, sub//': error setting offset time type')
    date = esmf_datedecrement(date, off, rc)
    call chkrc(rc, sub//': error decrementing current date')
  end if
end if

get_curr_calday = esmf_dategetfltdayofyear(date, rc)
call chkrc(rc, sub//': error return from esmf_dategetfltdayofyear')
```

1.1.13 is_end_curr_day (Source File: *time_manager.F90*)

Return true if current timestep is last timestep in current day.

INTERFACE:

```
function is_end_curr_day()
```

CONTENTS:

```
call get_curr_date(yr, mon, day, tod)
is_end_curr_day = (tod == 0)
```

1.1.14 is_end_curr_month (Source File: *time_manager.F90*)

Return true if current timestep is last timestep in current month.

INTERFACE:

```
function is_end_curr_month()
```

CONTENTS:

```
call get_curr_date(yr, mon, day, tod)
is_end_curr_month = (day == 1 .and. tod == 0)
```

1.1.15 is_first_step (Source File: *time_manager.F90*)

Return true on first step of initial run only.

INTERFACE:

```
function is_first_step()
```

CONTENTS:

```
nstep = esmf_timemgrgetnstep(tm_id, rc)
call chkrc(rc, sub//': error return from esmf_timemgrgetnstep')
is_first_step = (nstep == 0)
```

1.1.16 is_first_restart_step (Source File: *time_manager.F90*)

Return true on first step of restart run only.

INTERFACE:

```
function is_first_restart_step()
```

CONTENTS:

```
is_first_restart_step = first_restart_step
```

1.1.17 is_last_step (Source File: time_manager.F90)

Return true on last timestep.

INTERFACE:

```
function is_last_step()
```

CONTENTS:

```
is_last_step = esmf_tiememgrlaststep(tm_id, rc)
call chkrc(rc, sub//': error return from esmf_tiememgrlaststep')
```

1.1.18 timemgr_write_restart (Source File: time_manager.F90)

Write information needed on restart to a binary Fortran file. It is assumed that this routine is called only from the master proc if in SPMD mode.

INTERFACE:

```
subroutine timemgr_write_restart(ftn_unit)
```

CONTENTS:

```
call esmf_tiememgrrestartwrite(tm_id, rst_type, rst_nstep, rst_step_days, rst_step_sec, &
                               rst_start_ymd, rst_start_tod, rst_stop_ymd, rst_stop_tod, rst_ref_tod,
                               rst_curr_ymd, rst_curr_tod, rc)
call chkrc(rc, sub//': error return from esmf_tiememgrrestartwrite')

write(ftn_unit, iostat=rc) rst_type, rst_nstep, rst_step_days, rst_step_sec, &
                           rst_start_ymd, rst_start_tod, rst_stop_ymd, rst_stop_tod, rst_ref_tod,
                           rst_curr_ymd, rst_curr_tod

if (rc /= 0 ) then
  write (6,*) 'WRITE iostat= ',rc,' on i/o unit = ',ftn_unit
  call endrun
end if
```

1.1.19 timemgr_read_restart (Source File: time_manager.F90)

Read information needed on restart from a binary Fortran file. It is assumed that this routine is called only from the master proc if in SPMD mode.

INTERFACE:

```
subroutine timemgr_read_restart(ftn_unit)
```

CONTENTS:

```
read(ftn_unit, iostat=rc) rst_type, rst_nstep, rst_step_days, rst_step_sec, &
                           rst_start_ymd, rst_start_tod, rst_stop_ymd, rst_stop_tod, rst_ref_
                           rst_ref_tod, rst_curr_ymd, rst_curr_tod

if (rc /= 0 ) then
  write (6,*) 'READ iostat= ',rc,' on i/o unit = ',ftn_unit
  call endrun
end if
```

1.1.20 chkrc (Source File: *time_manager.F90*)

Checks the return code for errors

INTERFACE:

```
subroutine chkrc(rc, mes)
```

CONTENTS:

```
if ( rc == esmf_success ) return
write(6,*) mes
call esmf_errprint(rc)
call endrun
```